(Preprint) AAS 17-274

LAUNCH VEHICLE ASCENT TRAJECTORY SIMULATION USING THE PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES II (POST2)

Rafael A. Lugo,* Jeremy D. Shidner,* Richard W. Powell,* Steven M. Marsh,† James A. Hoffman,† Daniel K. Litton,‡ and Terri L. Schmitt§

The Program to Optimize Simulated Trajectories II (POST2) has been continuously developed for over 40 years and has been used in many flight and research projects. Recently, there has been an effort to improve the POST2 architecture by promoting modularity, flexibility, and ability to support multiple simultaneous projects. The purpose of this paper is to provide insight into the development of trajectory simulation in POST2 by describing methods and examples of various improved models for a launch vehicle liftoff and ascent.

INTRODUCTION

Trajectory simulation is a fundamental component of flight mechanics performance analyses, and many trajectory simulation tools are used in government, industry, and academia. In particular, the Program to Optimize Simulated Trajectories II (POST2) has been continuously developed for over 40 years and has been used in dozens of flight and research projects. The purpose of this paper is to provide insight into the development of trajectory simulation software by modeling a launch vehicle liftoff and ascent trajectory in POST2. A description of new POST2 features and improvements that have been recently implemented, as well as an analysis of the resultant simulation improvements, will be presented.

Mission Overview

The liftoff trajectory modeled in the present analysis is that of the Space Launch System (SLS). SLS is a heavy-lift launch vehicle designed to send crew and cargo to the Moon, Mars, asteroids, and beyond. The present work focuses on the liftoff and ascent trajectory of a crewed SLS Block 1B launch vehicle, scheduled to be flown for the crewed Exploration Mission 2 (EM-2) lunar flyby in 2021, as well as Europa and asteroid redirect missions respectively in 2022 and 2026. The SLS Block 1B configuration is shown in a detailed view in Figure 1. Existing Space Shuttle RS-25 engines and modified solid rocket boosters (SRBs) are used on the Core Stage. The upper portion of the vehicle consists of the Exploration Upper Stage (EUS), Orion Multi-Purpose Crew Vehicle (MPCV), and Launch Abort System (LAS).[1]

* Aerospace Engineer, Analytical Mechanics Associates, Inc., Hampton, VA 23666
† Software Engineer, Analytical Mechanics Associates, Inc., Hampton, VA 23666
‡ Aerospace Engineer, NASA Langley Research Center, Hampton, VA 23666
§ Aerospace Engineer, NASA Marshall Space Flight Center, Huntsville, AL 35812

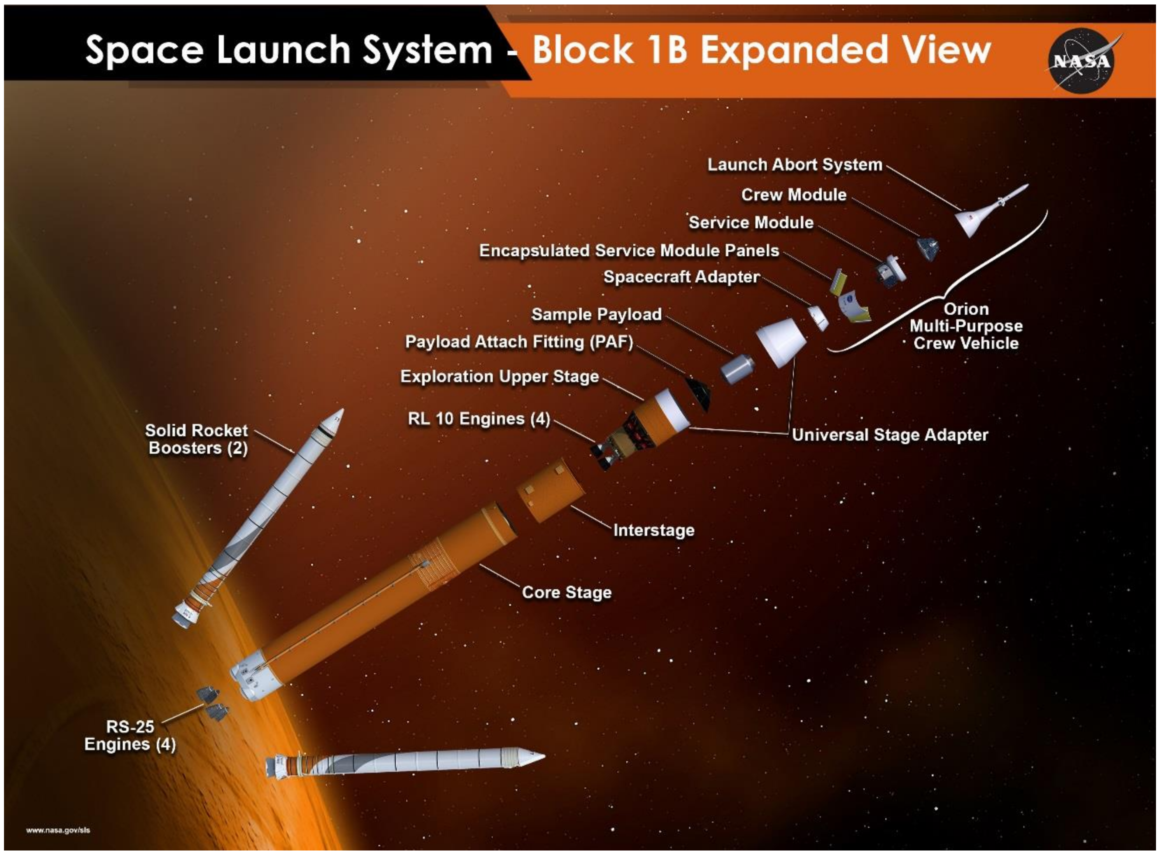

Figure 1. Space Launch System – Block 1B expanded view.[1]

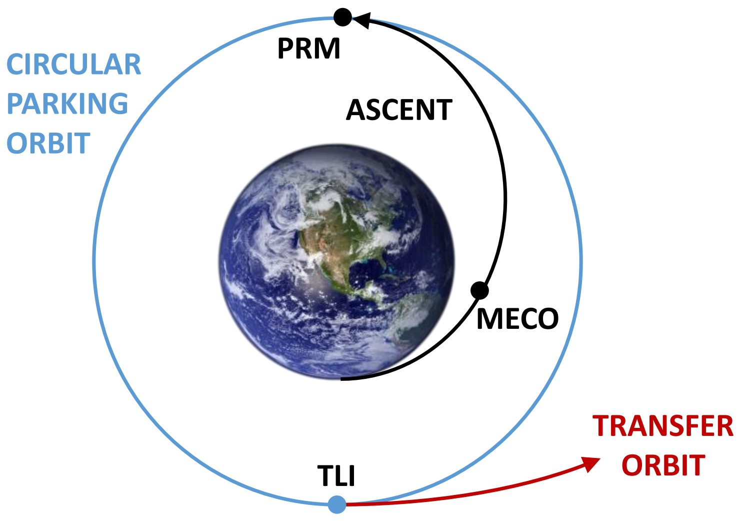

Figure 2. SLS Block 1B concept of operations (not to scale).

The ascent concept of operations is illustrated in Figure 2. SLS is launched from Cape Canaveral and ascends to Main Engine Cutoff (MECO), after which the core stage is jettisoned. The upper stage continues to coast until it reaches apoapsis, where the EUS performs the perigee raise maneuver (PRM) that puts the vehicle into a 100 nm altitude circular parking orbit. The vehicle remains in this parking orbit while the crew performs checkout procedures. The EUS then performs the Translunar Injection (TLI) burn before being jettisoned itself.[2] The current POST2 simulation models this trajectory until just after the end of the circularization burn performed by the EUS. However, there is capability in POST2 to continue beyond the TLI burn to lunar orbit, and return to splashdown on Earth.

PROGRAM TO OPTIMIZE SIMULATED TRAJECTORIES II

POST2 is an event-driven, point-mass trajectory simulation software with discrete parameter targeting and optimization capability. It provides multiple degrees-of-freedom (DOF) simulation and assessment of endo-and exo-atmospheric trajectories about a planetary body. The POST2 simulation capability includes, but is not limited to, launch, orbital, and entry phases of flight, vehicle design, development, and flight operations, as well as single and multiple vehicles working independently or in tandem. POST2 can solve a variety of flight mechanics and orbital transfer problems at multiple levels of fidelity. Low-fidelity problems, such as those for preliminary mission and vehicle design, may be completely input-driven and require no user-provided code or models. Higher-fidelity problems, such as those for flight operations, may utilize detailed vehicle models and flight code (e.g., guidance algorithms, parachute trigger logic, sensor models, etc.).

One of the primary functions of POST2, as its name implies, is to optimize trajectories, as well as to target user-specified conditions. To that end, POST2 contains a suite of features and tools to permit the user to design and control various types of optimization problems. POST2 includes implementations of a projected gradient method and a nonlinear program method by the Stanford Systems Optimization Laboratory[3] (PGM and NPSOL, respectively) for solving optimization problems.

Martin Marietta first delivered POST in 1973 to NASA Langley Research Center (LaRC) to support the Space Shuttle and other follow-on programs. POST was initially a 3-DOF tool, but it was continually improved over the following years. POST2 was delivered by Lockheed Martin (LM) in 1997 and provided many significant upgrades such as the capability to simulate and track multiple independent vehicles and switching between 3-DOF and 6-DOF equations of motion in a single simulation. LM ended active POST2 development in 2010. The POST2 development team continued at LaRC and distributed versions to other organizations, the most recent of which is POST2v3.[4] Though originally written as a combination of Fortran and C, the POST2 source code has since been rewritten primarily in C.

Since 2013, LaRC and the NASA Engineering and Safety Center (NESC) have committed to improving the POST2 architecture and implementing best practices in simulation software development and testing. To that end, POST2 has recently been re-architected as a new, streamlined product called POST2v4 that features improved modularity, flexibility, and is better able to support multiple simultaneous projects. The POST2 development team modifies, updates, and maintains POST2v4 to support a wide variety of flight and research projects, flight dynamics analyses, and end-to-end simulations. For the remainder of this paper, this new software will be referred to simply as POST2 unless there is a need to differentiate from POST2v3.

Architecture and Development Philosophy

The POST2 architecture has adopted a more modular and flexible approach that better addresses the needs of multiple simultaneous projects. POST2 consists of a “Core” code located in a centralized, version-controlled repository. Core contains “sockets” (in the form of C function pointers) into which users may “plug in” project-specific models and algorithms. Thus, projects no longer begin with a copy of the POST2 source code that was used for a previous flight project, as the project source code is kept and maintained separate from Core. A simplified diagram illustrating this Core/project code relationship is shown in Figure 3.

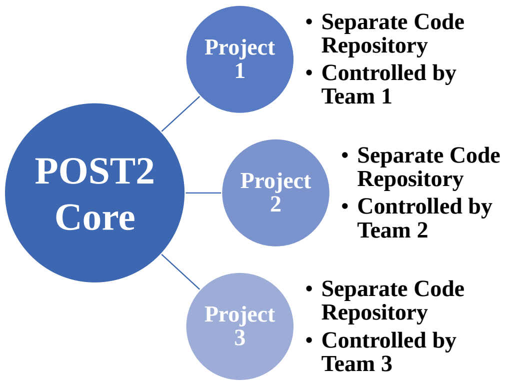

Figure 3. POST2 Core and project interface.

The POST2 Core software is the minimum set of algorithms and models required to simulate a trajectory. Core contains the code that interprets input files, integrates the equations of motion, and governs interfaces between various models. While it contains basic models of Earth gravity and atmosphere, it is generally up to the user to provide more detailed models such as those for vehicle aerodynamics, sensors, flight software, and higher-fidelity gravity and atmosphere.

Figure 4 shows an overview of the POST2 architecture. At left are the potential models that may interface with Core (this list is not comprehensive). At right, the user controls the simulation using the input file.

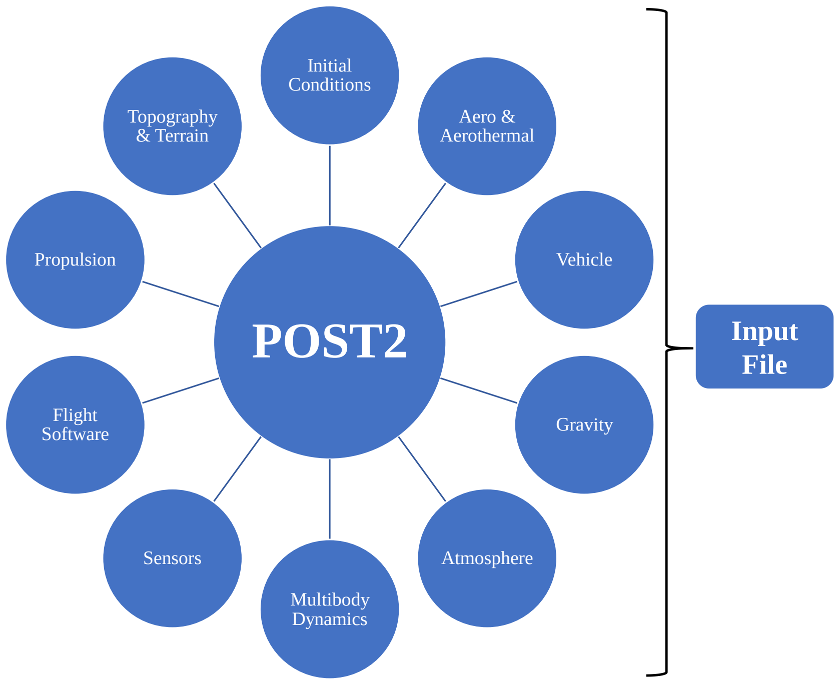

Figure 4. POST2 architecture.

The POST2 development philosophy has evolved in tandem with the architecture. Previously, a user with a new model to integrate with POST2 had to modify Core source code files; that is, they had to rewrite Core to interface with their model. The result was that the user had a modified version of POST2v3 that was custom to their project, which made it more difficult to debug, transfer, and maintain. POST2v4 utilizes C function pointers to provide interfaces through which users may implement models for aerodynamics, atmosphere, motion, etc. The user may still modify Core code by overriding files in Core with custom versions, but it is expected that this practice will occur less often as Core supports more projects and becomes more robust.

Another component of the POST2 development philosophy is the continual maintenance and support of project software, or continuous integration. Since the NESC began involvement in POST2 in 2013, it has required that flight simulations must be maintained through the life of the project or program. Thus, simulations may exist "on the shelf" for years, but must be ready to be compiled and run to respond to issues as they arise. The result is that LaRC has accumulated more than 20 simulations that are currently in use, in development, or archived. Each simulation is regression tested as Core is updated, which promotes (but does not ensure) survivability. These simulations include, but are not limited to, SLS 6-DOF liftoff, SLS 3-DOF liftoff and ascent[5], ISS, LDSD[6], LAS[7], InSight[8], and OSIRIS-REx[9].

Simulation Input Control

There are a variety of input controls available to the user that form the building blocks of a POST2 trajectory simulation. Of these, three of the most fundamental are the input file (also referred to as an input deck), events, and tables. The latter two are read by the program via the input file.

POST2 simulations are controlled by the user through the use of an ASCII input file that lists all data, events, equations, logic, and flags that dictate how the program interprets and processes that data. While the input file syntax is not standardized (e.g., it is not C or a Bash shell), it does utilize common programming logic, rules, and keywords. The user accesses and assigns data to POST2 variables through alphanumeric strings that are defined in Core or project dictionary header files. Conditional logic (if-then-else) statements are supported, so that equations, data, and/or events may be conditionally evaluated.

Input files follow a general structure. First, global simulation flags and constants, such as type of integrator, vehicle state initialization frame, and integration step size are declared. Event sequence and trigger criteria are then defined. Targeting and optimization flags, constraints, variables, and objectives are defined. Vehicle and mission models are also defined, usually in the first event. Finally, all subsequent events defining the rest of the trajectory are described. While this input structure is not required (i.e., all inputs related to an event, including its criteria, can be input together and not separated as indicated above), this approach has evolved from best POST2 programming practices followed over the years.

POST2 is event-driven, meaning that vehicle trajectories are described by a series of possible events. The first event usually contains the simulation and vehicle model definitions and global flags, and the final event terminates the simulation. Each event contains data that usually applies some change to the vehicle dynamics, and is triggered by specific criteria. For example, an event may describe a rocket ignition that starts 10 s after the previous event occurs, or when a certain altitude or velocity is reached.

One of the most fundamental POST2 constructs that facilitates simulation design and control is the data table. Data tables are commonly used as data input, and may be used to define the atmosphere, aerodynamics, thrust profiles, etc. Tables may be constant, monovariate, or multivariate (up to as many dimensions as needed). Available interpolation methods include step, linear, log, or cubic spline, and extrapolation is permitted. POST2 automatically evaluates tables given their independent variable (or variables) to provide the resultant value to the program. For example, the user may input a multivariate table that provides the vehicle drag coefficient as a function of angle of attack and Mach number. The drag coefficient may then be included by POST2 in its calculation of the forces acting on the vehicle. Tables may also be cross-referenced to avoid duplication of lengthy identical tables. For example, if a vehicle has two identical solid rocket motors, rather than input two separate propulsion tables that are also identical except in name, the user may input the table for the first engine and "refer" the table for the second engine to that of the first.

Simulation Features and Models

In general, a feature is added by the POST2 development team if it is determined to be useful to multiple projects. The feature is generalized (if needed) and integrated into Core with permission of the feature authors or owners. Other features are added by the development team as general improvements to functionality and ease of maintenance. This section will describe some of the new features and models relevant to the SLS Block 1B 3-DOF liftoff and ascent simulation.

Input Parser. The input parser is a processing system that interprets the input file, assigns data, and passes instructions to the program. A new C-based input parser has been implemented in POST2v4 that is easier to maintain, is better encapsulated, follows a more object-oriented design, and improves input file processing speed. Although the input file syntax remains unchanged, there is better enforcement of unambiguous input handling. The new input parser also supports variable aliasing, which permits the user to assign alternate names to POST2 variables at the input file level as opposed to the source code data dictionary header level for improved readability.

Functional Equation Parser. The functional equation parser is a new POST2 feature that permits the user to define and evaluate equations directly from the input file using POST2 input variables, requiring no modifications to source code. The resultant outputs are also POST2 variables that are recognized by the program and may be used as event triggering criteria, optimization variables, etc. The functional equation parser enables the design and implementation of models such as simple steering and control laws. Thus, the user may create a model directly in the input file, facilitating fast model design and testing before developing higher-fidelity models.

Function Pointers. The ability to integrate custom models and algorithms without modifying Core is provided through the use of function pointers. Function pointers (referred to earlier in this paper as "sockets") are entry points located throughout Core through which the user may hook in their project-specific modules as needed. Forms of modules existed in previous versions of POST2, but were limited in scope and flexibility. The introduction of function pointers in POST2 advances the concept of modules such that they may be included or omitted at compile time, and managed from the input file. Thus, the user may add multiple modules that model the same system, and then toggle between which module to use for a given run. The driving philosophy is that not all compiled POST2 builds need to include all capabilities. For example, an SLS liftoff simulation does not need to include Mars terrain models within its build. Thus, the implementation of models through function pointers means that unnecessary capabilities may be omitted from compilation without causing POST2 to fail. As previously discussed, this also means that project-specific code may be kept and maintained separately from Core.

The following features and models are not new to POST2v4, but are critical to the present simulation problem.

Guidance and Steering. A variety of guidance and steering options that define how the vehicle orientation changes or is maintained are available to the user through the input file. For example, the vehicle attitude may be specified using inertial and relative aerodynamic and Euler angles, inertial pitch plane angles, sun clock and cone angles, etc. Steering angle profiles between events may be specified using tables, cubic polynomials, piece-wise linear functions, etc.

Generalized Acceleration Steering. Generalized acceleration steering (GSA) is an instantaneous and iterative steering option in which the specified steering equations are solved on each integration pass, meaning that steering criteria are continuously satisfied. GSA utilizes projected gradient methods for targeting and pseudo-inverse techniques to permit solving under-constrained problems. Example GSA uses include attitude trimming a vehicle in roll, pitch, and yaw with gimbaled engines, and maintaining steady level flight by determining the throttle setting and angle of attack that causes the derivatives of velocity and flight path angle to be zero. Other uses include the ability to handle constraints specified by angles in multiple Euler angle sequences. Note that GSA may only be used with constraints that are directly affected by the controls at the same instant of time or integration step (instantaneous as noted above). For example, changing thrust to target a velocity will not work with GSA as that is a time-integrated effect; that is, any change to thrust at a given instant in time will not affect vehicle velocity at that time.

TRAJECTORY SIMULATION IMPROVEMENTS

Previous versions of the simulation have been implemented in SLS Block 1 end-to-end mission architecture design and planning and more generalized Mars mission architecture design.[5,10] In the present analysis, the SLS Block 1B liftoff and ascent trajectory is simulated using 3-DOF equations of motion. Vehicle mass properties and aerodynamics are described by data tables and values provided by the SLS Program. Atmosphere is also described by a table. Gravity is modeled using the built-in POST2 spherical harmonic model up to the J8 term. The optimization constraints (dependent variables) include SRB and Core Stage fuel use, Core Stage disposal limits, and maximum dynamic pressure, angle of attack, and g-loads. Minimum times between jettison events and subsequent engine burns are specified. A circular orbit at the end of MECO and a specified C3 at the end of the upper stage TLI burn is also targeted. The optimization controls (independent variables) include launch azimuth, pitch profile control variables, burn durations, payload weight at launch, and propellant offload. The optimization problem is to maximize the payload weight at the end of TLI using the projected gradient method.

Guidance and Steering

Gravity Turn. As previously mentioned, GSA permits the user to specify constraints in multiple Euler angle sequences. A gravity turn after tower clear is implemented by using GSA to solve for angles of attack, sideslip, and roll that maintains a vehicle heads-down orientation. A gravity turn requires that the angle of attack and sideslip angle be zero, while the heads-down configuration requires a constant 180° roll angle. Thus, the requirements are in two different Euler angle sequences: bank-sideslip-angle of attack and yaw-pitch-roll. GSA determines the bank angle that will produce a constant 180° roll angle. This is shown in Figure 5. The solid lines show that with GSA, a constant roll angle is held. The dashed lines show that without GSA, roll angle drifts while bank angle stays constant.

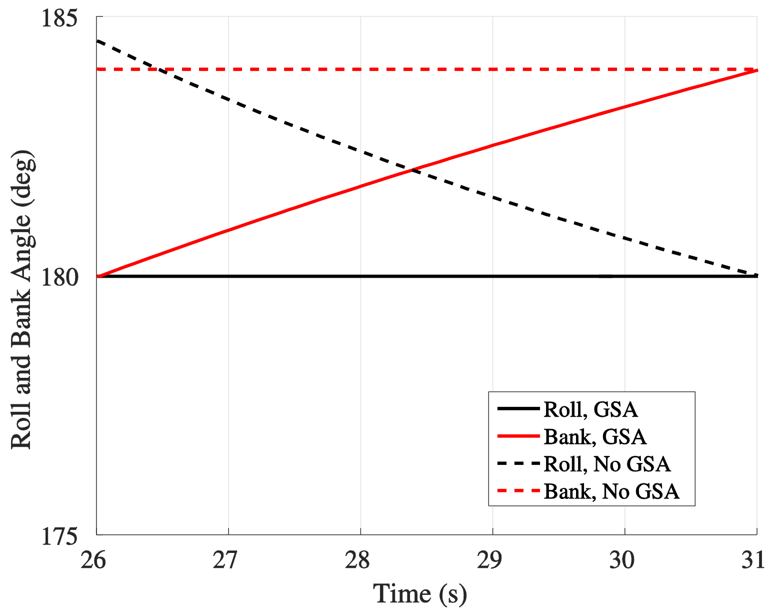

Figure 5. Comparison of gravity turn initialization with and without GSA.

Linear Tangent Steering. The nominal pitch profile of the vehicle during ascent is commanded using polynomial steering in POST2. The pitch profile between each event takes the form

$$\theta = a + bt \tag{1}$$

where θ is the instantaneous vehicle inertial pitch angle, t is the variable of which the polynomial is a function (usually time, but can be any POST2 variable), a is the current inertial pitch angle, and b is the pitch rate variable (if t is time) that is usually left to be optimized by POST2. To design the pitch profile using this approach, the user must set up multiple events, each of which are triggered by timers and in which the constant b in Eq. (1) is optimized by POST2. The initial guesses of b at each of these events are the pitch rates. This is not a particularly intuitive approach to designing a pitch profile, since the pitch rates are usually very small and depending on the number of pitch events, it can be difficult to compute accurate gradients for pitch events.

To demonstrate the flexibility of the POST2 functional equation parser, the nominal pitch profile of the coast between MECO and PRM was replaced by linear tangent steering, which is a steering law in which a unique set of three constants completely define the altitude, velocity, and flight path angle at the end of a finite burn.[11] The linear tangent steering law is described by

$$\tan\theta = \tan\theta_0 - \left(\tan\theta_0 - \tan\theta_f\right)\frac{t}{t_f} \tag{2}$$

where θ is again the instantaneous vehicle inertial pitch angle, θ_0 is the pitch angle at the start of the steering event, θ_f is the desired pitch angle at the end of the steering event, t is the instantaneous time, and t_f is the time at the end of the steering event.

The linear tangent law in Eq. (2) was implemented in the simulation input file using the functional equation parser, and the final time and final pitch angle were set as independent variables in the POST2 optimization problem. No changes to source code were required; the entire steering law was implemented from the input file. Figure 6 shows the steering law response in the inertial pitch angle for the coast between MECO and PRM. The pitch profiles look similar, as expected, since the improvement is in the user being able to specify the initial guesses of final pitch angle and length of steering segment, rather than a non-intuitive pitch rate as in Eq. (1).

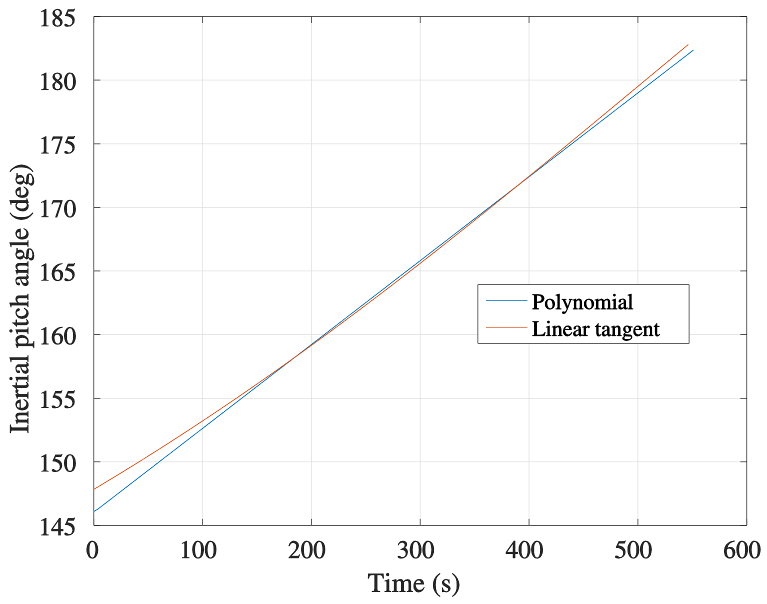

Figure 6. Comparison of polynomial and linear tangent steering laws applied to coast segment between MECO and PRM. The initial time marks the beginning of the steering segment.

Propulsion

Pseudo Actuator. Traditionally, POST2 utilizes third-order polynomial command profiles to adjust vehicle throttle, Euler angles, etc., such that the vehicle responds to a continuous function. For a throttle system that ramps up to a rate and down to the final command, the user must program three phases in POST2:

1. Use polynomial profiles to ramp up the quantity to its rate limit,
2. Hold the polynomial at a constant rate, and
3. Ramp the polynomial down to the target value.

This setup is not intuitive since it involves multiple event targets and non-intuitive use of polynomials. Thus, POST2 users have often reduced this complexity by assuming instantaneous throttle acceleration, resulting in an instantaneous change in the rate (i.e., a discontinuity in the derivative) at the ramp up and ramp down that does not capture the behavior of the real system.

In an effort to make this modeling more realistic, a “pseudo actuator” module has been implemented in the simulation that uses an input command value to change a quantity subject to rate and acceleration limits. The new response generated by the pseudo actuator model better approximates the response of a real control system that is subject to rate and acceleration limits.

The simplicity of the pseudo actuator model eliminates the need for multiple POST2 events and non-intuitive use of polynomials since the desired value is now changing relative to an input command, and rate and acceleration limits. The command is instead set in a single POST2 event and the simulation progresses without need to create the constant rate or ramp down events. Instantaneous rates of change are also avoided. Additionally, the pseudo actuator model may be applied to variables for which a ±180° discontinuity may exist.

Figure 7 shows the pseudo actuator module applied to the SLS throttle system as it ramps up to 109% throttle. The blue line shows the traditional throttling method where an instantaneous rate of change is seen at the ramp up and ramp down, the red line shows the new pseudo actuator model subject to rate and acceleration limits, and the green line is the response of the representative model of the actual throttle system used in the 6-DOF SLS GNC simulation..

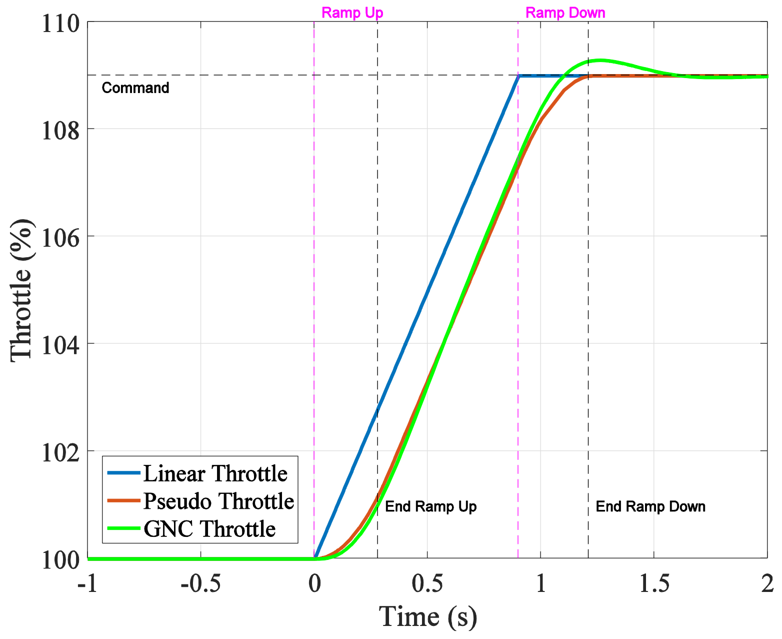

Figure 7. Comparison of throttle responses.

Independent Fuel and Oxidizer Flow Rates. It is becoming increasingly common for modern engine designers to specify engine flow rates per component, rather than more traditional methods of specifying total fuel and oxidizer mixture ratio per engine. Therefore, POST2 now has the ability to implement flow rates that are independent of mixture ratio. This capability was achieved by implementing a generalized tank model in POST2 that permits the user to define a combination of oxidizer, fuel, and any other flow rates directly. The flow rates may be assigned to engines, between tanks, or as overboard venting. This model is similar to past POST2 models of propellant tanks, with the distinction that tank-to-tank flow and multi-flow engines are now possible. Thus, the user now has the ability to book-keep usage of specific flows during vehicle operation, as opposed to lumping all flows to fit into the traditional definitions of fuel and oxidizer.

Outputs

In addition to the normal POST2 output data files, two additional types of outputs have been added to the SLS Block 1B liftoff simulation.

JSON Output. The ability to output data in JavaScript Object Notation[12] (JSON) format using POST2 function pointers has been added to facilitate communication between POST2 and Copernicus for SLS end-to-end (E2E) optimization work. Time histories of POST2 variables, selected by the user at the input file level, are written to a file in the JSON format.

Visualization. Although POST2 does not provide any trajectory or analysis visualization tools, it is able to interface with visualization software such as Exploration Visual Environment (EVE).[13] EVE is a simulation, visualization, and analysis system designed to integrate engineering data with a virtual environment to support mission design, planning, and analysis. It integrates time-dependent data with detailed graphical models within a full-scale three-dimensional solar system or independent reference frame, enabling the user to gain valuable insight into correlation of data with simulation events. A POST2 module has been implemented that writes ASCII data files containing trajectory data in a format natively readable by EVE, resulting in a seamless transition between trajectory simulation and visualization.

Figure 8 and Figure 9 show screenshots of visualization output using EVE, right after liftoff and during the TLI burn. 3-DOF state information is provided directly by the POST2 simulation output. In Figure 8, the red, maroon, yellow, and blue geometries at left represent the tower and lightning protection system keep-out zones. In both figures, the cyan line is the SLS CG track. The red cones at the base of the SLS and EUS are thruster plumes. In Figure 9, the EUS is not to scale.

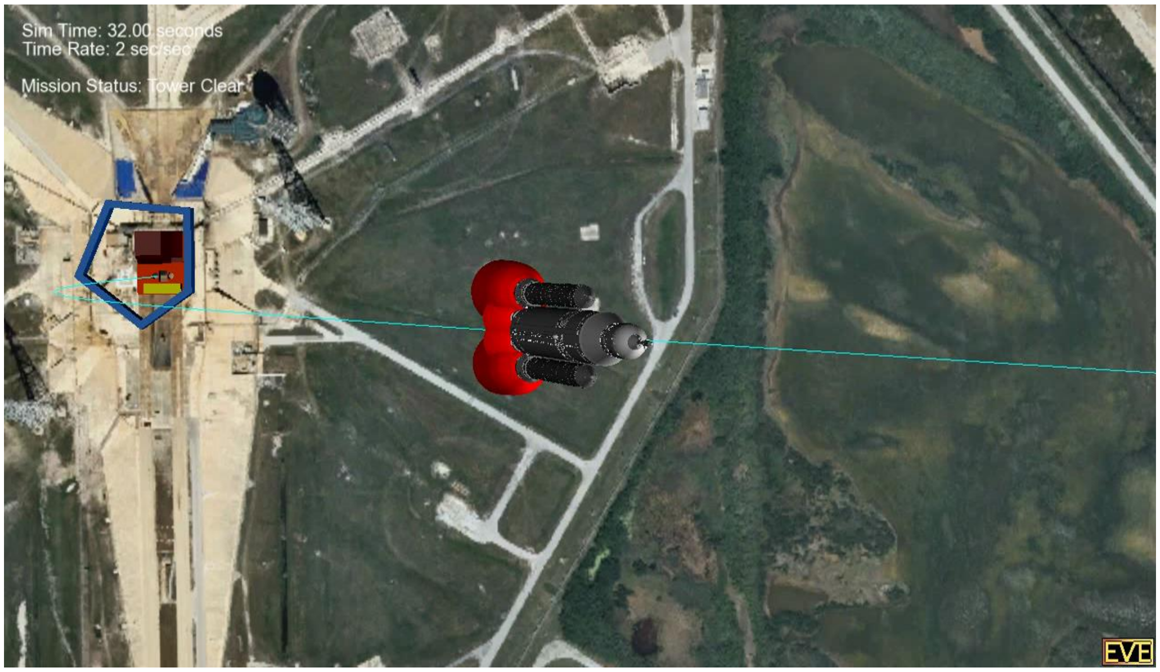

Figure 8. Image of SLS liftoff after tower clear obtained from EVE animation.

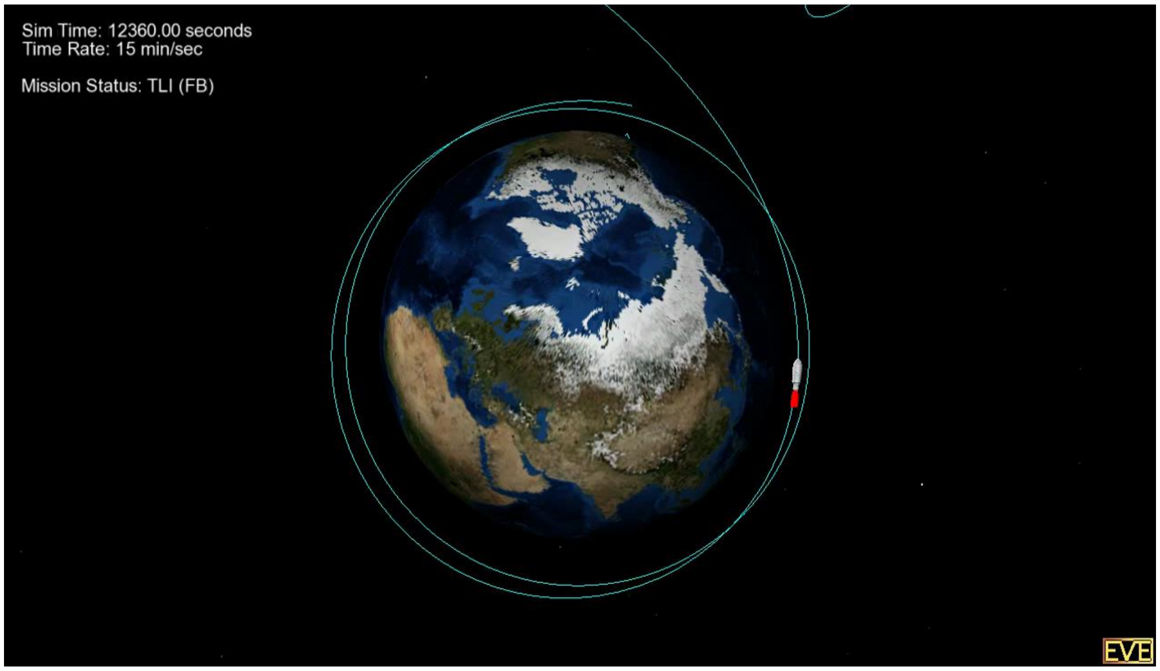

Figure 9. Image of TLI burn obtained from EVE animation.

SUMMARY

Updates and new capabilities added to the POST2 program and architecture have been presented. New features and improvements to POST2 modularity and flexibility were highlighted by demonstrating improvements to the SLS Block 1B liftoff and ascent 3-DOF trajectory in POST2. These improvements include implementation of a gravity turn using GSA, implementing a linear tangent steering segment using the functional equation parser, using the pseudo actuator feature to improve modeling of engine throttle response, adding independent fuel and oxidizer rate specification to more closely model the provided fuel consumption data, and adding the capability to automatically generate JSON output files and EVE data files for visualization.

These improvements allow the user to explore the design space faster and more efficiently, while also permitting better understanding and characterization of the vehicle response early in the design cycle. Additionally, while these new features were demonstrated here in a 3-DOF sense, many of them are also be used in 6-DOF simulations. Ultimately, the new POST2 architecture and evolving design philosophy helps position the user to better respond to flight mechanics simulation problems and analyses.

ACKNOWLEDGEMENTS

The authors thank Scott Craig (Marshall Space Flight Center) for SLS trajectory information, John Aguirre and Carole Garrison (AMA, Inc.) for POST2 software development and support, and James Beaty, Angela Bowes, Alicia Cianciolo, Jody Davis, Soumyo Dutta, Justin Green, Eric Queen, Paul Tartabini, Matthew Toniolo, Carlie Zumwalt (LaRC), Joseph White, and Richard Winski (AMA, Inc.) for POST2 model development and additional software support. The authors also thank Jill Prince (NESC), Joseph Gasbarre, Robert Maddock, Ronald Merski, Scott Striepe, and David Way (LaRC) for POST2 architecture planning and programmatic support.

NOTATION

DOF	Degree of Freedom
EM	Exploration Missions
EUS	Exploration Upper Stage
GSA	Generalized Acceleration Steering
JSON	JavaScript Object Notation
LaRC	NASA Langley Research Center
LAS	Launch Abort System
LDSD	Low Density Supersonic Decelerator
LM	Lockheed Martin
MECO	Main Engine Cutoff
MPCV	Multi-Purpose Crew Vehicle
NESC	NASA Engineering Safety Center
NPSOL	Nonlinear Program Solver
PGM	Projected Gradient Method
POST2	Program to Optimize and Simulate Trajectories II
PRM	Perigee Raise Maneuver
SLS	Space Launch System
SRB	Solid Rocket Booster
TLI	Trans-Lunar Injection

REFERENCES

[1] "Illustration of Expanded View of the Block 1B Configuration," NASA, URL: https://www.nasa.gov/sites/default/files/thumbnails/image/sls_block_1b_poster-expanded.jpg, [cited 11 January 2017].

[2] "NASA's First Flight With Crew Will Mark Important Step on Journey to Mars," NASA, URL: https://www.nasa.gov/feature/nasa-s-first-flight-with-crew-will-mark-important-step-on-journey-to-mars, [cited 17 January 2017].

[3] "NPSOL," Stanford Business Software Inc., URL: http://www.sbsi-sol-optimize.com/asp/sol_product_npsol.htm, [cited 17 January 2017].

[4] "Program to Optimize Simulated Trajectories II," NASA Langley Research Center, URL: https://post2.larc.nasa.gov/, [cited 11 January 2017].

[5] Litton, D. K., et al., "Creating an End-to-End Simulation for the Multi-Purpose Crewed Vehicle," AAS 15-641.

[6] Bowes, A., et al., "LDSD POST2 Simulation and SFDT-1 Pre-Flight Launch Operations Analyses," AAS 15-232.

[7] Litton, D. K., et al., "Reverse Launch Abort System Parachute Architecture Trade Study," AIAA 11-1225.

[8] "InSight Mars Lander," NASA, URL: https://www.nasa.gov/mission_pages/insight/overview/index.html, [cited 17 January 2017].

[9] Gal-Edd, J., and Cheuvront, A., "THE OSIRIS-REX Asteroid Sample Return – MISSION Operations Design," 13th International Conference on Space Operations, Pasadena, California, 5-9 May 2014.

[10] Lugo, R., et al., "A Robust Method to Integrate End-to-End Mission Architecture Optimization Tools," 2016 IEEE Aerospace Conference, 5-12 March 2016, Big Sky, MT.

[11] Perkins, F. M., "Derivation of Linear-Tangent Steering Laws", Nov. 1966, Air Force Report No. SSD-TR-66-211.

[12] "The JSON Data Interchange Format," 1st Edition, Ecma International, Standard ECMA-404, October 2013.

[13] "Exploration Visual Environment User's Guide," Analytical Mechanics Associates, Inc., Version 2.11.